Orchestrating BMD Control in Extended BPEL

Thomas S. Cook
Computer Science Department
Naval Postgraduate School
Monterey, CA, USA
tscoo1@nps.edu

Duminda Wijesekera
Computer Science Department
George Mason University
Fairfax, VA, USA
dwijesek@gmu.edu

Bret Michael
Computer Science Department
Naval Postgraduate School
Monterey, CA, USA
bmichael@nps.edu

Man-Tak Shing
Computer Science Department
Naval Postgraduate School
Monterey, CA, USA
shing@nps.edu

Abstract – *We specify duty cycles of a Ballistic Missile Defense (BMD) command and control application by decorating the Business Process Execution Language (BPEL) with Quality of Service (QoS), Measures of Performance (MoP), Measures of Effectiveness (MoE) and Measures of Merit (MoM) metrics.*

Key Phrases: Web Services, Business Process Execution Language (BPEL), service-oriented architecture, Command, Control, Communication, Intelligence, Surveillance, and Reconnaissance (C4ISR), Quality of Service (QoS), MoE, MoP.

1 Introduction

Command, Control, Communication, Intelligence, Surveillance, and Reconnaissance (C4ISR) applications require making decisions based on situational awareness created by fusing sensory information collected from independently maintained sources. Having a command and control (C2) structure that respects the autonomy of basic services facilitates the flexibility to dynamically negotiate and adjust to changes in the battle space while maintaining the continuity of the overall operations and deployment readiness. In this paper we develop such a framework to thwart threats from ballistic missiles by using a three-tiered C2 structure. This sits well with the U.S. DoD's objective of adopting Service Oriented Architecture (SOA) in which the master orchestrator provides a service by composing the services of the autonomously functioning sub-services. The continuity of orchestrated operations is modeled by duty cycles, with each duty cycle reacting to environmental changes. The orchestrator provides the required quality of service (QoS) – which includes timeliness as one aspect [1]. As shown in this paper, a flexible QoS-sensitive SOA suffices to specify and implement stated C4ISR requirements.

The DoD mandated the basic web services (WS) framework standards for use in Enterprise Resource Planning (ERP) software packages, but has not mandated the standards for use in the Global Information Grid (GIG) as the standards fall short in meeting GIG security and authorization requirements [2]. The basic WS framework standards include what are commonly referred to as the *core* web service development standards; Web Services Description Language (WSDL), SOAP, and Universal Description, Discovery and Integration (UDDI). Although the core WS have been applied successfully by industry in business systems, as Birman et al. claim [3], they fall short of C4ISR needs due to the lack of support for time-critical events. Consequently, in this study we decorate BPEL specified duty cycles with QoS, specifically timeliness attributes, MoP and MoE specifications, with the hope that a SOA satisfying the need articulated by Birman et al. can implement our design. The rest of the paper is organized as follows. Section 2 specifies use cases for a ballistic missile defense system. Section 3 presents an overview of conventional C2 and possible execution using WS and specifies a C2 family of WS using WSDL, and Section 4 presents their process integration using BPEL. Section 5 discusses the evolution of the Operations Order. Section 6 concludes the paper.

2 BMD C2

The objective of the Missile Defense Agency's (MDA) Advanced Battle Manager (ABM) of the Ballistic Missile Defense System (BMDS) [5] is to provide an integrated, layered defense from ballistic missiles of all ranges in all phases of their flight. At a high level, the BMDS consists of an integrated C2, Battle Management (BM), and Communications (collectively known as C2BMC), and weapons and sensors. Weapons and sensors are capable of engaging and sensing many different threat missiles through different phases of their flight: boost, mid-course and terminal. The C2 component is responsible for creating and distributing the operations orders (OPORD), that essentially provides initial weapons, and sensor locations, their orientations, and their responsibilities within the plan while the BM executes the battle according to the OPORD and the responses from sensory inputs.

Wijesekera, Michael and Nerode [4] use three kinds of agents to model BMDS C2: the strategic commander agent (SCA), regional commander agent (RCA), and the tactical commander agent (TCA). Each battle manager assumes one of these roles. A hierarchical command structure in [4] consists of SCAs at the top of the C2 structure that share information horizontally between them.

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **21 MAY 2008**	2. REPORT TYPE **N/A**	3. DATES COVERED **-**

4. TITLE AND SUBTITLE **Orchestrating BMD Control in Extended BPEL**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Computer Science Department Naval Postgraduate School Monterey, CA, USA**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
AFCEA-GMU C4I Center Symposium "Critical Issues In C4I" 20-21 May 2008, George Mason University, Fairfax, Virginia Campus

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**UU**	**13**	

Each SCA manages vertically down the chain to its assigned RCA's and with any assigned sensors and weapons in the sensors and weapons nets. Continuing down the C2 structure the individual RCA's manage and communicate with their assigned TCAs and any assigned weapons and sensors in the weapons and sensors nets. Finally, each TCA manages and communicates with its assigned weapons and sensors within the sensors and weapons nets. While information travels up and down the C2 structure, most down-flows are commands and most up-flows are status reports.

2.1 Assumptions

1. Operations Orders (OPORDs) have been issued by all agents and each agent has established its defenses. This implies that all weapon and sensor systems for the entire BMDS are positioned to defend against the most likely threat missile attack according to the OPORD intelligence estimate; weapon systems have specific orientations ready to launch and sensors are in surveillance mode.
2. Given that the threat attacks according to the OPORD intelligence estimate, we assume the weapon and sensor systems execute the plan autonomously with little or no interference from the command agents.
3. In our scenario the threat does not attack according to the intelligence estimate and SCA1, RCA2 and assigned TCAs must manage the initial attack to defeat the threats.

2.2 Scenario execution

Our scenario shown in Figure 1 proceeds as follows. RCA2's organic sensor and BM determine that three separate threat missiles are inbound and predicted to hit a high-priority asset on its Prioritized Defended Asset List (PDAL). According to the OPORD the terminal-defense mission for the asset being attacked is assigned to TCA22 and the midcourse defense of the asset is assigned to TCA21. However, based on the OPORD Intel annex most resources in SCA1's area of operation are oriented on the enemy's likely air avenue of attack, depicted by the large dotted arrow and labeled as such in the figure. Therefore, reorienting of resources within RCA2's area of operation is necessary to negate the threats. RCA2 concurrently sends a contact message to SCA1 requesting permission to reorient resources and engage the threats, in addition to sending a be-prepared-to-launch order to TCA21 and TCA22.

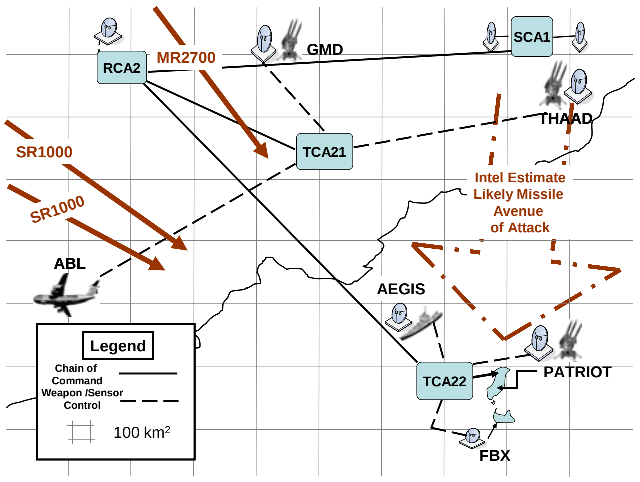

Figure 1 BMDS Scenario

The messages being exchanged include the individual threat tracks and values for specific QoS attributes, MOP, and MOE necessary to ensure the threats are engaged prior to reaching their keep-out ranges. In this scenario RCA2 has two MOEs associated with it: (i) survivability, defined as the fraction of defended assets that survive the attack and (ii) the probability that the interceptor kills the threat target given that it arrives in time. The MOP associated with RCA2 is Time-on-Target; the time remaining for any weapon system to launch an interceptor. This is included to give subordinates an upper bound on time to engage with the appropriate shot doctrine.

TCA21 receives the be-prepared order (to launch) and steps into the kill chain cycle at the assign weapon task. Using the track information from RCA2, TCA21 determines the appropriate weapon systems with which to engage the threat missiles, builds an engagement plan, and issues a be-prepared order to the appropriate weapons and sensors. Likewise, TCA22 receives its be-prepared order to launch, but its launch is contingent on the threat reaching the keep-out range. TCA22 also steps into the kill chain cycle at the assign weapons task and issues be-prepared missions to its associated weapons and sensors.

Upon receiving SCA1's response message to launch, RCA2 issues a message to the TCAs to execute the be-prepared missions sent earlier. TCAs 21 and 22 use the MOEs and MOP from RCA2 to guide the selection of the services required to complete the weapons assignment, engagement, and assess kill tasks of the kill chain. We show this in detail in subsequent sections, but first we describe the scenario for each of the agents: SCA1, RCA2, and TCAs 21 and 22. We show, in use case (a technique for describing how to achieve a goal or task) format, the kill chain tasks performed by each agent in our scenario. Figure 2 shows the process logic a BM executes upon receiving a track list

Use Case 1: *Detect*
Goal in Context: Identify threats from a list of reported sensor tracks
Scope & Level: A primary task of the battle Manager
Preconditions: Battle manager has been initialized
Success End Condition: Correctly identify threat object.
Failed End Condition: Fails to identify threat missile.
Primary Actor: Battle Manager
Trigger: Receive track list from a sensor
MAIN SUCCESS SCENARIO
1. Receive Track List message
2. Verifies source of message
3. Validate request parameters
4. Associates Track List
5. Correlates Track List
6. Returns a threat list

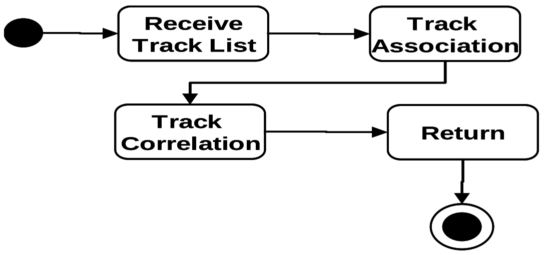

Figure 2: Detect

Figure 3 shows the execution logic of a BM upon receipt of a track list from a sensor in its C2 structure.

Use Case 2: *Track*
Goal in Context: Return a launch quality threat track
Scope & Level: A primary task of the battle Manager
Preconditions: Detect Task complete
Success End Condition: Produces fire quality tracks
Failed End Condition: Fails to produce fire quality track
Primary Actor: Battle Manager
Trigger: Receive a threat list
MAIN SUCCESS SCENARIO
1. Receive Threat List message
2. Verifies source of message
3. Validate request parameters
4. Fuse threat List
5. Calculate IPP for threats
6. Calculate Aim Point for threats
7. Calculate time available to kill threats
8. Calculate QoS, MOP, MOE requirements
9. Returns a threat list, QoS, MOP, MOE

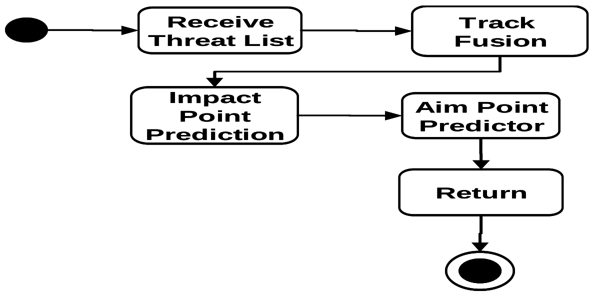

Figure 3: Track

Figure 4 shows the process logic a BM executes upon receiving a threat track list.

Use Case 3: *Assign Weapon*
Goal in Context: Identify the best weapon system to destroy the threat
Scope & Level: A primary task of a BM
Preconditions: Detect and track tasks in the kill chain has successfully completed
Success End Condition: Identify weapon to destroy the identified threat missiles.
Failed End Condition: A weapon system is not identified
Primary Actor: BM
Trigger: Receive a weapons assignment message
MAIN SUCCESS SCENARIO
1. Receive Launch message
2. Verifies source of message
3. Validate request parameters
4. Monitor QoS requirements
5. Identify available resources
6. Target weapon pairing
7. Return result

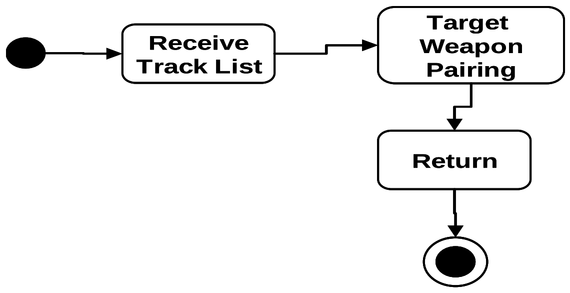

Figure 4: Assign Weapon

Figure 5 shows the process logic a BM executes upon receiving a threat track list with weapons assignment solution.

Use Case 4: *Engage*
Goal in Context: Assigned weapon launches interceptor
Scope & Level: This is a primary task of the TCA
Preconditions: TCA has been initialized
Success End Condition: TCA assigns Launch to appropriate (based on message) weapon or responds to caller that no weapon is available

Failed End Condition: TCA fails to assign weapon or report that there is a problem in launching to caller
Primary Actor: TCA
Trigger: Receive Launch from superior
MAIN SUCCESS SCENARIO
1. Receive Launch message
2. Verifies source of message
3. Validate request parameters
4. Monitor QoS Requirements
5. Build engagement plan
6. Send plan
7. Monitor QoS Requirements

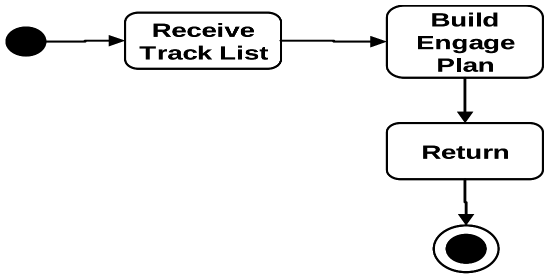

Figure 5: Engage

Figure 6 shows the process logic a BM executes upon receiving an assess kill message. Figure 7 shows the process logic a BM executes upon first receiving an initialization message followed some time later by an assign weapon message.

Conventional battle managers follow a duty cycle commonly referred to as a *kill chain* [5] consisting of the following tasks: detect, track, assign weapon, engage, and assess kill. The kill chain begins when a sensor reports an object to a BM agent. The agent continues to track the object while determining if the object poses a threat, and if the object does pose a threat, assigns an available interceptor to destroy it. After the firing of the interceptor, the BM agent continues to monitor and assess the engagement; if the initial interceptor fails to destroy the threat missile and the shot doctrine used dictates a second shot (e.g. shoot-look-shoot policy) the weapon system re-engages the threat with updated target information.

Use Case 5: *Assess Kill*
Goal in Context: Determine correctly the result of an engagement.
Scope & Level: A primary task of the TCA
Preconditions: TCA has been initialized
Success End Condition: TCA returns a correct assessment of an engagement
Failed End Condition: TCA fails to return a correct assessment of an engagement
Primary Actor: TCA
Trigger: Receive Launch from superior
MAIN SUCCESS SCENARIO
1. Receive Launch message
2. Verifies source of message
3. Validate request parameters
4. Monitor QoS Requirements
5. Report engagement result
6. Return result to caller

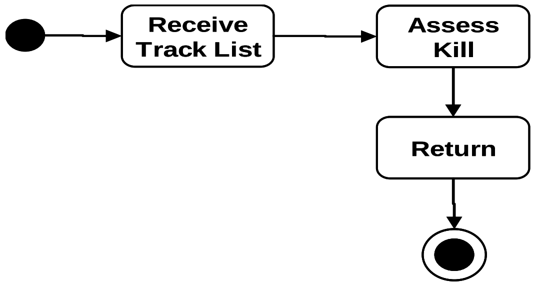

Figure 6: Asses Kill

In addition to the possibility of a weapons system missing a target there exists the possibility that a BM has no weapons systems available for assignment. In this situation the BM alerts its superior so that an alternative BM can be chosen for the mission; it is customary in military operations to have this built in to the plan and therefore the engage task would have planned to have a number of weapons systems and BM's on stand-by (be prepared mission) for these type of circumstances. After completion of the kill assessment the duty cycle repeats.

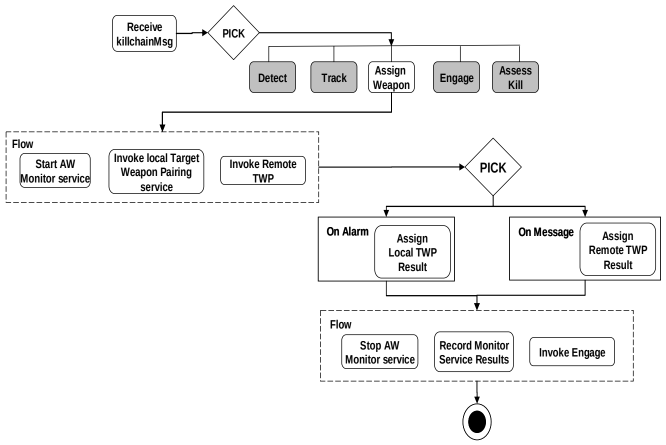

Figure 7: Assign Weapon Process

3 Web Services for battle management

We now specify a conventional BM as a service in a SOA by specifying the kill chain as a periodic process that is the main orchestrator of a BPEL process decorated with QoS, MOE, and MOP extensions specified in [14]. Selecting the participating partner services of the main kill chain is based on the client QoS and MOE parameters.

1. **Target Association Service (TA):** Begins the kill chain when it receives a list of potential tracks of the threat missile from the sensor net as reported by radars, using a track association algorithm to identify the track objects reported by these sensors.
2. **Track Correlation Service (TC):** Uses a correlation algorithm to compare the reduced track list against a known threat database to classify the missile. If the track object does not match, but observed measurements (e.g. its velocity is in the range of a ballistic missile) makes it suspicious, it is marked suspicious and assigned to additional sensors for observation. All others are logged for offline analysis.
3. **Track Fusion Service (TF):** Track objects gathered thus far are used to create an enhanced description.
4. **Impact Prediction Point Service (IPP):** Predicts the impact point of the threat missile.
5. **Aim Point Predictor Service (APP):** Computes an aiming point for each track object.
6. **Target Weapon Pairing Service (TWP):** Computes the most appropriate weapons systems to engage the threats.
7. **Engagement Planner Service (EP):** Output from TWP and information from the Operations Plan (OPLAN) is used by EP to design, issue, and notify all parties of the plan to destroy the threat missiles.
8. **Assess Kill Service (AK):** Assess battle damage using the sensor net to complete the entire kill chain cycle.

The detect task is composed of the TA and TC services. The track task is composed of TF, IPP, and APP services. The assign weapons task is composed of TWP service. Engage task is composed of an EP service and the assess kill task consists of the AK service. We list the QoS, MoP and MoE parameters of each of the eight services and the five tasks in Table 1. Figure 9 shows the detect task as a composition of the selected TA service and the TC service.

```
<portType name="DetectPT">
  <operation name="DetectProcess">
    <mopList>
      <mop name="ExecutionTime" value="12sec"/>
      <mop name="Accuracy" value=".998"/>
    </mopList>
    <moeList>
      <moe name="DetectTargetInBOOST" value="null"/>
    </moeList>
    <input message="tns:DetectMsgRequest"/>
    <output message="tns:DetectMsgResponse"/>
  <operation>
</portType>
```

Figure 8: Detect Composition WSDL

Task/Service	**QoS**	**MoP**	**MoE**
Kill Chain	Availability Reliability	Execution Time Accuracy	Kill threat prior keep out range
Detect	Availability Reliability	Execution Time Accuracy	Detect target in boost
Track	Availability Reliability	Execution Time Accuracy	Monitor track without loss of contact
Assign Weapon	Availability Reliability	Execution Time Accuracy	Assign best weapon available
Engage	Availability Reliability	Execution Time Accuracy	Create best mission to destroy threat and monitor BDA
Assess Kill	Availability Reliability	Execution Time Accuracy	Assign best sensor to conduct BDA
Track Association (**TA**)	Availability Reliability	Execution Time Accuracy	Identify and assoc correct number of tracks with its source
Track Correlation (**TC**)	Availability Reliability	Execution Time Accuracy	ID track Objects as threat
Track Fusion (**TF**)	Availability Reliability	Execution Time Accuracy	Enhance threat object by fusing data from multiple sensors
Impact Point Prediction (**IPP**)	Availability Reliability	Execution Time Accuracy	Determine IPP within 10 m^2
Aim Point Prediction (**APP**)	Availability Reliability	Execution Time Accuracy	Determine Aim Point within 100 cm^2
Target Weapon Pairing (**TWP**)	Availability Reliability	Execution Time Accuracy	Best weapon to kill target
Engage Planner (**EPS**)	Availability Reliability	Execution Time Accuracy	Best plan to destroy target prior to keep out range
Assess Kill (**AK**)	Availability Reliability	Execution Time Accuracy	Best sensor to conduct BDA

Table 1: QoS, MOP, MOE

```
<portType name="TrackAssocPT">
  <operation name="AssocTrackList">
    <mopList>
      <mop name="ExecutionTime" value="5sec"/>
      <mop name="Accuracy" value=".999"/>
    </mopList>
    <moeList>
      <moe name="AssocTrackToSource" value="null"/>
    </moeList>
  <input message="tns:TrackAssocMsgRequest"/>
  <output message="tns:TrackAssocMsgResponse"/>
 <operation>
</portType>
```

Figure 9: Track Association WSDL

```
<portType name="TrackCorrPT">
 <operation name="CorrTrackList">
  <mopList>
   <mop name="ExecutionTime" value="5sec"/>
   <mop name="Accuracy" value=".999"/>
  </mopList>
  <moeList>
   <moe name="IDThreatGivenThreat" value="NULL"/>
  </moeList>
  <input message="tns:TrackCorrMsgRequest"/>
  <output message="tns:TrackCorrMsgResponse"/>
 <operation>
</portType>
```

Figure 10: Track Correlation WSDL

Finally, we show the WSDL of a complete kill chain that is composed of the higher level tasks detect, track, assign weapon, engage, and assess kill which are themselves composed of the atomic level services described earlier.

```
<portType name="KillChainPT">
 <operation name="killThreat">
  <mopList>
   <mop name="ExecutionTime" value="27sec"/>
   <mop name="Accuracy" value=".95870"/>
  </mopList>
  <moeList>
   <moe name="Defend_Asset" value="NULL"/>
  </moeList>
  <input message="tns:KillChainMsgRequest"/>
  <output message="tns:KillChainMsgResponse"/>
  <fault name="Fail" message="FailNotice:"/>
 <operation>
</portType>
```

Figure 11: Kill Chain WSDL

In each composition instance execution time was the MOP used to select a service.

One of the two extensions necessary to orchestrate C4ISR is the need for QoS sensitivity, for which we use the lightweight Q-WSDL extension in [13]. In particular, we use the Operational Latency class where execution times of every operation are specified. The second is the use of the shadow pattern of [11] that specifies exception handling. We use message types, messages and services with the standard notations of ‘*’ for zero or more repetitions, ‘?’ for zero or one repetitions, and ‘+’ for one or more repetitions.

3.1 Message Types

Tables 2 and 3 list sample basic and complex WSDL data types [9] used in exchanged messages.

Type Name	**Primitive**	**Example**
myId	Long Int	123456789245
sensorID	Long Int	454656736363
Availability	Boolean	Yes/No
weaponName	String	THHAD, AEGIS, …
sensorName	String	FBX, SBX, …
ammoStatus	String	Green, Red, Yellow
timeToEngage	Duration	P0y0m0dt0h0m3s
dateTimeGroup	DateTime	2007-05-31T13:20:00-05:00
Hostile	Boolean	Yes/No
Latitude	Long Int	765468642222
Longitude	Long Int	367463823982
Velocity	Long Int	645646455467
Acceleration	Long Int	832678326864

Table 2: Basic types of message elements

Type Name	**Type Structure**	**Example**
OPORD	xmlns:OPORD="http://swe.nps.edu/BMDS/OPORD	opord20080129
Track	XmlNS=URI#trackType	Track Structure
Weapon	XmlNS=URI#weaponType	Weapon structure
Sensor	XmlNS=URI#sensorType	Sensors structure
Sca	XmlNS=URI#scaType	sca structure
Tca	XmlNS=URI#tcaType	tca structure
QoS	XMlNS=URI#qwsdl:operationType [13]	QoS Structure
Bond	XmlNS=URI#Time	$3.5 Cred 1
Turing test	Image	10101..01

Table 3: Complex types of message elements

3.2 Messages

A sample, assignWeaponMsg, is shown in Listing 1. Other domain-specific messages are listed in Table 4 below with their definitions. Similarly Listings 2 and 3 show other control messages. Finally, Tables 5 and 6 describe some services provided by the TCA and other third parties.

1. **<message** name="**AssignWeaponMsg**">
2. <part name="ID" element="message ID"/>
3. <part name="Track" element="string"/>
4. <part name=”OPORD” element=”OPORD”/>
5. <part name="DATE" element="Time"/>
6. <part name="QOS" element="QoSType"/>
7. <part name=”surety” element=”Bond”/>
8. <part name=”Ack” element=”wantAck”/>
9. <part name=”Sign” element=”PKISignature/>*
10. <part name=”RTT reply” element=”Turing test

```
</message>
```

Listing 1: WSDL AssignWeaponMsg

Message Type	**Utility**
detectMsg	Kill chain task to associate tracks with a source and determine if the track is a threat
trackMsg	Kill chain task to fuse track information, determine the threat impact point, and calculate an aim point
assignWeaponMsg	Kill chain task to assign the most appropriate weapon to negate a know threat
engageMsg	Kill chain task to build an engagement plan to defeat a threat
assessKillMsg	Kill chain task to monitor engagement and report Battle Damage Assessment
launchInterceptorMsg	Command to Launch an interceptor
cancelLaunchMsg	Command to cancel a previous launch command
weaponHSMsg	Command to return the health and status of all weapons

Table 4: Types of Messages

```
<message name="initializeBMMsg">
<part name="id" element="long"/>
<part name="”OPORD0129312008”
element=”OPORD”
</message>
<message name="deRequisitionMsg">
<part name="ID" element="long"/>
</message>
```

Listing 2: WSDL Application Data

```
<message name="FailNotice">
<part name="Date" element="dateTime"/>
<part name="ID" element="long"/>
<part name="ERROR" element="string"/>
<part name="Sign_PKI" element="string"/>
</message>
<message name="LaunchInterceptorReciept">
<part name="DATE" element="dateTime"/>
<part name="ID" element="long"/>
<part name="comment" element="string"/>
<part name="Sign_PKI" element="string"/>
</message>
```

Listing 3: WSDL Control Data

```
<portType name="assignWeaponPT">
<operation name="assignWeapon">
<input message="tns:assignWeaponMsg"/>
<output message="tns:assignWeaponReciept"/>
<faultname="faultassignWeapon "message="tns:FailNotice" / >
</operation>
</portType>
```

Table 5: WSDL Port Type Specs for BM services

```
<portType name="monitorServicePT">
<operation name="monitor">
<input message="tns:startMsg"/>
<output message="tns:StartNotificationMsg"/>
<fault name="monitorfault"
message="tns:FailNotice"/>
</operation>
</portType>
<portType name="timerPT">
<operation name="startTimer">
<:input message="tns:startMsg"/>
</operation>
</portType>
```

Table 6: WSDL Port Type for C2 Third Party Services

3.3 Operations Order (OPORD)

An OPORD is, “a directive issued by a commander to subordinate commanders for the purpose of effecting the coordinated execution of an operation” [16]. The OPORD is a vital document in ballistic missile defense as it explains in detail the responsibilities of all systems. The OPORD, at a minimum, contains unit task organization and the five paragraphs of (1) Situation (2) Mission (3) Execution (4) and Service Support (5) Command and Signal.

In traditional land warfare combat commanders issue their orders to subordinate commanders who in turn prepare and issue orders to their subordinates until each combatant in every unit knows his or her mission and the mission of those two levels up the chain of command. The initial OPORD of nearly all campaigns are routinely more detailed and well thought out than subsequent OPORDs. This tendency is a direct reflection of the amount of time available to plan prior to hostilities beginning. For the initial order, units may have days, weeks, and even months to plan and issue the orders. Once hostilities begin, the time to plan generally decreases and makes the development, issuance and coordination of plans more difficult, in addition to reducing timelines to days or hours.

In missile defense the timelines are significantly shorter than traditional land warfare combat scenarios discussed above. In the missile defense domain timelines can be in the range of several minutes to as little as 30 seconds.

With such short timelines we look to perform autonomous execution of missile defense engagements where we remove the human from the loop. For this reason the OPORD must be designed to be read and “understood” by computers; we accomplish this in our case study by constructing our OPORDs using the Resource Description Framework (RDF)[15]. RDF is a W3C Recommendation for describing Web resources and is designed to be read by computers. We show in Table 7 below our OPORD written in RDF/XML for the scenario described above and pictured in Figure 1. The RDF provides the means to describe the complex structure of the OPORD so that it can be understood by the participating BMs. In Table 7, lines 40, 44, 48, 52, and 56 show the five minimum essential paragraphs of an OPORD as defined in [16]. Each of the five paragraphs is a property that has a reference to a resource containing information about the particular property. As an example we show at line 47 the property OPORD:MISSION has a reference to a resource containing information about the TCA’s MISSION; the text of an actual mission for TCA22 is in bold. It is certain that some of the other paragraphs have sub-graphs and each of those can be defined by a value or as in the case of the OPORD MISSION a reference to another resource.

```
1.  <?xml version="1.0"?>
2.  <rdf:RDF xmlns:rdf=
3.  "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
4.  xmlns:rdfs="http://www.w3.org/2000/01/rdf-
            schema#"
5.  xmlns:OPORD="http://swe.nps.edu/BMDS/OPORD
            #">
//The rdf description element that describes our
resource
// OPORD
6.  <rdf:Description rdf:about=
7.  "http://swe.nps.edu/BMDS/OPORD/opord20080129
            ">
8.  <OPORD:classification>
9.  <rdf:Alt>
10. <rdf:li>UNCLASS</rdf:li>
11. </rdf:Alt>
12. </OPORD:classification>
13. <OPORD:CopyNumOfNumCopies>1 of 100
14. </OPORD:CopyNumOfNumCopies>
15. <OPORD:issuingHQ>TCA21</OPORD:issuingHQ
            >
16. <OPORD:placeofIssue>452143</OPORD:placeofIs
            sue>
17. <OPORD:DTGSignature>012920080100
18. </OPORD:DTGSignature>
19. <OPORD:MsgRefNum>012920080245
20. </OPORD:MsgRefNum>
21. <OPORD:OrderNumber>01292008-45
22. </OPORD:OrderNumber>
23. <OPORD:codeName>Butkus</OPORD:codeName>
24. <OPORD:references>
25. <rdf:Seq>
//Reference to the SCA’s Initiating OPORD Code
Name Lambert
26. <rdf:li rdf:resource=
27. "http://swe.nps.edu/BMDS/documents/OPORD/Lam
            bert">
28. </rdf:li>
//Reference to the RCA2’s Initiating OPORD Code
Name HAM
29. <rdf:li rdf:resource=
30. "http://swe.nps.edu/BMDS/documents/OPORD/Ham
            ">
31. </rdf:li>
32. </rdf:Seq>
33. </OPORD:references>
34. <OPORD:timeZoneUsed>ZULU
35. </OPORD:timeZoneUsed>
//Task Organization is defined by the resource URI
below
36. <OPORD:taskOrginization
37. rdf:resource=
38. "http://swe.nps.edu/BMDS/OPORD/opord20080129
39. /AnnexATaskO"></OPORD:taskOrginization>
//The first of the minimum essetial elements of the
five
//Paragraph operations order; SITUATION
40. <OPORD:SITUATION
41. rdf:resource=
42. "http://swe.nps.edu/BMDS/OPORD/opord20080129
43. /Situation"></OPORD:SITUATION>
//The second of the minimum essetial elements of the
five
//Paragraph operations order; MISSION
44. <OPORD:MISSION
// The URI at line 46 is a reference to the document
that
//contains the following mission statement for this
OPORD
//MISSION: 060004282008 (Z) TCA22 forces Defend
//assets according to the Priority Defended Assets
List
//(PDAL) against anticipated ballistic missile attacks
//within region.
45. rdf:resource=
46. "http://swe.nps.edu/BMDS/OPORD/opord200801
            29
// The URI at line 48 is a reference to the document
that
//contains the next Higher level of commands (RCA2
in
this scenario) mission statement for this OPORD
//MISSION: 060004282008 (Z) RCA2 forces Defend
//assets according to the Priority Defended Assets
List
```

```
//(PDAL) against anticipated ballistic missile attacks
//within region.
47. <OPORD:HIGHERMISSION
48. rdf:resource= " http://swe.nps.edu/BMDS/documents
/OPORD/Ham.MISSION
//The third of the minimum essetial elements of the
five
//Paragraph operations order; EXECUTION
49. <OPORD:EXECUTION
50. rdf:resource=
51. "http://swe.nps.edu/BMDS/OPORD/opord20080129
52. /Execution"></OPORD:EXECUTION>
//The fourth of the minimum essetial elements of the
five
//Paragraph operations order; SERVICE
SSUPPORT
53. <OPORD:SERVICESUPPORT
54. rdf:resource="http://swe.nps.edu/BMDS/OPORD
55. /opord20080129/ServiceSupport">
56. </OPORD:SERVICESUPPORT>
//The fifth of the minimum essetial elements of the
five
//Paragraph operations order;
COMMANDANDSIGNAL
57. <OPORD:CMDSIGNAL
58. rdf:resource="http://swe.nps.edu/BMDS/OPORD
59. /opord20080129/CommandSignal">
60. </OPORD:CMDSIGNAL>
61. <OPORD:CDRSNAMERANK>COOKGEN
62. </OPORD:CDRSNAMERANK>
63. <OPORD:AUTHENNAMEPOS>PULFORD2IC
64. </OPORD:AUTHENNAMEPOS>
//ANNEXES A-Z OF THE OPORD EACH IS A RDF
//RESOURCE WHOS DESCRIPTION IS FOUND
AT
//THE APPROPRIATE URI
65. <OPORD:ANNEXES>
66. <rdf:Seq>
67. <rdf:li
            rdf:resource="http://swe.nps.edu/BMDS/OP
            ORD
68. /opord20080129/AnnexATaskO"></rdf:li>
69. …<rdf:li
            rdf:resource="http://swe.nps.edu/BMDS/OP
            ORD
70. /opord20080129/AnnexZDistro"></rdf:li>
71. </rdf:Seq>
72. </OPORD:ANNEXES>
73. <OPORD:DISTROrdf:resource="http://swe.nps.edu
74. /BMDS/OPORD/opord20080129/AnnexZDistro">
75. </OPORD:DISTRO>
76. </rdf:Description>
</rdf:RDF>
```

Table 7: Operations Order

4 BPEL Orchestration of BM

In this section, we specify the TCA using BPEL [10], where the TCA Assign Weapon Process invokes necessary local and remote services. In Table 8, TCA is activated upon receiving an initialization message, which includes an operations order shown in Table 7, from the RCA, to establish the organizational structure thereby creating the chain of command for the SCA, RCA, TCA and weapons and sensors nets. Once the TCA process completes initialization it blocks waiting for one of the predefined messages detect, track, assign weapon, engage, assess kill, cancel launch, switch mode, or other commands from its RCA. In our scenario TCA21 receives the assign weapon message from RCA2 line numbers 47-49 of Table 8.

As specified in lines 50-67, in response to an assign weapon message, the TCA invokes a local monitoring service, to record information on the executing services and the assign weapons task in its entirety and the remote target weapon pairing (RTWP) service algorithm. At line 110 the process invokes a local synchronous Local Target Weapon Pairing (LTWP). This service acts as a shadow [11] to the RTWP service.

If the assign weapon process does not receive a result from the RTWP within 10 seconds, an alarm is triggered in line 40 alerting the process to use the result from the LTWP service for the rest of the task.

Upon receiving the TWP result the process invokes the Engage task line 148 with the result and waits 10 seconds for a callback, line 156, signaling that the engage task has been initiated after which the process invokes the stop monitor and records the QoS, MOP, and MOE results. Finally, if the process does not receive the callback message from the engage task it invokes the warning callback line 136 to the calling client signaling that the task completed, but there is no evidence that the engage task received the results or has begun execution.

```
1.  <?xml version="1.0" encoding="UTF-8"?>
2.  <:process
3.  xmlns:AW="http://swe.nps.edu/bmds/services/AW"
4.  :ENG="http://swe.nps.edu/bmds/services/ENGAGE"
5.  ...
6.  <import importType="http://schemas.xmlsoap.org/wsdl/"
    location="WSDL/AwMonitor.wsdl"
    ns="http://swe.nps.edu/BMDS/service/awonitor/">
7.  …<partnerLinks>
8.  <:partnerLink myRole=
9.  "awService" name="assignWeapon"
    partnerLinkType="AW:awLT"
    partnerRole="AwCustomer"/>
10. …</:partnerLinks>
11. <:variables>
12. <:variable messageType="AW:AWMsg"
```

```
    name="AWMsg"/>
13. …</:variables>
14. <:flow>
15. <:links>
16. …</:links>
// lines 17 – 19 Receive Assign Weapon from RCA2
// execute the targetWeaponPairing operation
17. <:receive createInstance="yes"
18. name="ReceiveAWMsg"
19. operation="targetWeaponPairing"
    partnerLink="assignWeapon"
    portType="AW:assignWeaponPT" variable="AWMsg">
20. <:sources>
21. <:source linkName="L3"/>
22. </:sources>
23. </:receive>
24. <:pick name="PickLocalOrRemoteResult">
25. <:targets>
26. <:target linkName="L9"/>
27. </:targets>
28. <:sources>
29. <:source linkName="L4"/>
30. </:sources>
// line 31-35 if receive the callback message assign the
// results to engageMsg1
31. <:onMessage operation="TwpCallback"
    partnerLink="remoteTwpLT"
    portType="rtwp:TwpCallbackPT"
    variable="remoteTwpResponseMessage">
32. <:assign name="AssignRemoteTWPResult">
33. <:copy>
34. <:from part="result"
    variable="remoteTwpResponseMessage"/>
35. <:to part="awResult" variable="engageMsg1"/>
36. </:copy>
37. </:assign>
38. </:onMessage>
39. <:onAlarm>
40. <:for>PT10S</:for>
41. <:assign name="AssignLocalTWPResult">
42. <:copy>
43. <:from part="result"
44. variable="localTWPResponseMessage"/>
45. <:to part="awResult" variable="engageMsg1"/>
46. </:copy>
47. </:assign>
48. </:onAlarm>
49. </:pick>
// lines 50-67 CONCURENTLY Call the remote target
// weapon pairing algorithms and the moitoring service
50. <:flow name="FlowStartMon_RemoteTWP">
51. <:targets>
52. <:target linkName="L3"/>
53. </:targets>
54. <:sources>
55. <:source linkName="L5"/>
56. </:sources>
57. <:links>
58. <:link name="L1"/>
59. <:link name="L2"/>
60. </:links>
61. <:invoke inputVariable="startMonitorRequestMessage"
    name="InvokeAWMonitorService"
    operation="startMonitor" partnerLink="monitorLT"
    portType="awmon:startMonitorPT">
62. <:targets>
63. <:target linkName="L1"/>
64. </:targets>
65. </:invoke>
66. <:invoke inputVariable="remoteTwpRequestMessage"
    name="InvokeRemoteTWP" operation="Twp"
    partnerLink="remoteTwpLT"
    portType="rtwp:remoteTwprequestPT">
67. <:targets>
68. <:target linkName="L2"/>
69. </:targets>
70. </:invoke>
71. <:assign name="AssignMonitorParams">
72. <:sources>
73. <:source linkName="L1"/>
74. </:sources>
75. <:copy>
76. <:from>
77. <:literal>start</:literal>
78. </:from>
79. <:to part="start"
    variable="remoteTwpRequestMessage"/>
80. </:copy>
81. </:assign>
82. <:assign name="PassTrackList">
83. <:sources>
84. <:source linkName="L2"/>
85. </:sources>
86. <:copy>
87. <:from>
88. <:literal>tracklist</:literal>
89. </:from>
90. <:to part="start"
91. variable="remoteTwpRequestMessage"/>
92. </:copy>
93. </:assign>
94. </:flow>
95. <:sequence name="SequenceLocalTWP">
96. <:targets>
97. <:target linkName="L5"/>
98. </:targets>
99. <:sources>
100.<:source linkName="L6"/>
101.</:sources>
102.<:assign name="PassTrackList">
103.<:copy>
104.<:from>
105.<:literal>start</:literal>
106.</:from>
```

```
107.<:to part="start" variable="localTWPRequestMessage"/>
108.</:copy>
109.</:assign>
// line 110 synchronous call to the local target weapon
// paring algorithm
110.<:invoke inputVariable="localTWPRequestMessage"
    name="InvokeLocalTWP" operation="localTWP"
    outputVariable="localTWPResponseMessage"
    partnerLink="localTWPLT"
111.portType="ltwp:localTWPPT"/>
112.</:sequence>
113.<:pick name="Wait10SecForCallback">
114.<:targets>
115.<:target linkName="L8"/>
116.</:targets>
// line 117 waiting for callback from the Engage task that
// is invoked after Assign weapon completes its task.
117.<:onMessage operation="callback"
    partnerLink="Engage"
    portType="ENG:engageCallBackPT"
    variable="callBackMsg">
118.<:flow>
119.<:links>
120.<:link name="L10"/>
121.</:links>
// line 122 – 127 Stop the monitor and record the results
122.<:invoke inputVariable="stopMonitorRequestMessage"
    name="InvokeStopMonitor" operation="stopMonitor"
    partnerLink="StopMonitorLT"
    portType="awmon:stopMonitorPT">
123.<:sources>
124.<:source linkName="L10"/>
125.</:sources>
126.</:invoke>
127.<:receive name="RecordMonitorResponseResults"
    operation="monitorCallback" partnerLink="monitorLT"
    portType="awmon:monitorCallbackPT"
    variable="monitorResponseMessage">
128.<:targets>
129.<:target linkName="L10"/>
130.</:targets>
131.</:receive>
132.</:flow>
133.</:onMessage>
134.<:onAlarm>
135.<:until>P10S</:until>
// line 136  invokes callback alerting the client that while
// assign weapon completed its task it has not received
// confirmation from the engage task
136.<:invoke inputVariable="WarningMsg"
    name="InvokeWarningCallback"
137.operation="warningCallback"
    partnerLink="assignWeapon"
    portType="AW:WarningPT"/>
138.</:onAlarm>
139.</:pick>
// Line 140 Receive the results from the remote target
// weapon pairing algorithm
140.<:receive name="ReceiveRemoteTWPcallback"
    operation="TwpCallback" partnerLink="remoteTwpLT"
    portType="rtwp:TwpCallbackPT"
    variable="remoteTwpResponseMessage">
141.<:targets>
142.<:target linkName="L6"/>
143.</:targets>
144.<:sources>
145.<:source linkName="L9"/>
146.</:sources>
147.</:receive>
// line 148 invoke the Engage task of the kill chain
148.<:invoke inputVariable="engageMsg1"
    name="InvokeEngage" operation="engage"
    partnerLink="Engage" portType="ENG:engagePT">
149.<:targets>
150.<:target linkName="L4"/>
151.</:targets>
152.<:sources>
153.<:source linkName="L7"/>
154.</:sources>
155.</:invoke>
//156 receive a callback from the engage task alerting the
// client that task handoff is complete
156.<:receive name="ReceiveEngageCallback"
    operation="callback" partnerLink="Engage"
    portType="ENG:engageCallBackPT"
    variable="callBackMsg">
157.<:targets>
158.<:target linkName="L7"/>
159.</:targets>
160.<:sources>
161.<:source linkName="L8"/>
162.</:sources>
163.</:receive>
164.</:flow>
165.</:process>
```

Table 8 The TCA Process

5 Evolution of OPORDS

As discussed in Section 3.3 above the OPORD provides the Mission of two higher levels of command and tasks to subordinates. Once the initial order is received the commander must delete the higher level commands mission and add his own. In addition, the commander must remove the tasks to the subordinates and provide tasks to his subordinates. Typical information in the tasks might be things such as the defended sector assignment and orientation of weapons systems and asset to be defended in sector (e.g. from the PDAL).

The initial OPORD would have been sent during initialization prior to any of the messages received in Table 8 above. However, in our scenario the threat ballistic missiles do not attack according to the intelligence estimate sent out in the initial OPORD and RCA issues a FRAGMENTARY ORDER (FRAGO) as part of the AssignWeaponMsg to its subordinate TCAs to be prepared to reorient weapons systems and sensors in line 17. The AssignWeaponMSG would contain those parts of the OPORD that had change; for instance the mission is the same, but the execution paragraph would task sensors and weapons to reorient in the general direction of the incoming target so that the weapons systems could engage at the earliest opportunity.

6 Conclusions

We show how the BPEL with appropriate extensions for MoPs, MoEs and QoS parameters can be used to specify command, control, and battle management needs of Ballistic Missile Control. In follow on work we intend on showing a much more rigorous and complete design of the BM for the scenario proposed in this paper.

Acknowledgement

The research is funded in part by a grant from the U.S. Missile Defense Agency. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright annotations thereon.

References

[1] Raymond Paul, Jaideep Srivastava and Duminda Wijesekera, *Information Quality Based (Computer/Distributed) System Evaluation*, in International Journal of Testing and Evaluation, June 2000, pages 41-52.

[2] Transformation Focus - Situational Awareness *SOA in The DOD* http://www.army.mil/armybtkc/focus/sa/soa-dod.htm

[3] Ken Birman, Robert Hillman, Stefan Pleisch. Building Net-Centric Military Applications over Service Oriented Architectures. SPIE Defense and Security Symposium 2005. March 29-31, 2005. Orlando, Florida. http://www.cs.cornell.edu/projects/quicksilver/.

[4] Duminda Wijesekera, James B. Michael, and Anil Nerode. BMD Agents: An Agent-Based Framework to Model Ballistic Missile Defense Strategies. In Proc. 6th Int. Workshop on Policies for Distributed Systems and Networks, IEEE (Stockholm, Sweden, June 2005), pp. 115-118.

[5] Dale Scott Caffall. Developing Dependable Software for A System-Of-Systems. Ph. D. Dissertation, Naval Postgraduate School, Monterey, CA. March 2005.

[6] Missile Defense Agency. Global Ballistic Missile Defense: A Layered Integrated Defense. BMDS Booklet, fourth Edition www.mda.mil/mdalink/pdf/bmdsbook.pdf

[7] Alistair Cockburn, Use Case Template http://alistair.cockburn.us/index.php/Basic_use_case_template.

[8] S. Kaushik, D. Wijesekera and P. Amman, BPEL Orchestration of secure WebMail, Technical Report ISE-TR-06-08, George Mason University, Fairfax, VA, August 2006.

[9] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana, Web Services Description Language (WSDL) 1.1, 2001.

[10] A. Alves, et el., Business Process Execution Language, OASIS Standard, 11 April, 2007.

[11] T. W. Otani, M. Auguston, T. S. Cook, D. Drusinsky, J. B. Michael, and M. Shing, "A design pattern for using non-developmental items in real-time Java", Proceedings of the 5th international workshop on Java technologies for real-time and embedded systems (JTRES 2007), Vienna, Austria, September 26 - 28, 2007, pp. 135-143.

[12] Matthias Kloppmann, Et el. WS-BPEL Extension for People – BPEL4People, July 2005

[13] Andrea D'Ambbrogio, A Model-driven WSDL Extension for Describing the QoS of Web Services, International Conference on Web Services (ICWS'06), Chicago, USA, September 18-22, 0-7695-2669-1/06, IEEE, Computer Society

[14] U.S. Department of Defense. Department of Defense Dictionary of Military and *Associated Terms*. Joint Pub. 1-02, Apr. 12, 2001 (as amended through May 23, 2003).

[15] Dave Beckett, RDF/XML Syntax Specification (Revised) W3C Recommendation 10 February 2004, http://www.w3.org/TR/rdf-syntax-grammar/

[16] Headquarters Department of the Army, FM 5-0, Army Planning and Orders Production, Washington D.C., 20 January 2005, http://www.army.mil/usapa/doctrine/Active_FM.html

Thomas S. Cook is a Lieutenant Colonel in the US Army and a PhD candidate at the Naval Postgraduate School in Monterey California.

Duminda Wijesekera is an associate professor in the Department of Information and Software Engineering at George Mason University, Fairfax, Virginia. During various times, he has contributed to research in security, multimedia, networks, systems, avionics and theoretical computer science. These span topics such as applying logical methods to access and dissemination control, securing circuit switched (SS7) and IP based telecommunication (VoIP) systems, multimedia, security requirements processing during the early phases of the software life cycle, WWW security, railroad signaling security, SCADA security, communicating honeynet farms, and engineering Ballistic Missiles. His pre-GMU work has been in quality of service issues in multimedia, avionics control and specifying and verifying concurrent systems using logical methods.

He holds courtesy appointments at the Center for Secure Information Systems (CSIS) and the Center for Command, Control and Coordination (C4I) at George Mason University, The Computer Science Department at the Naval Postgraduate School, NIST, and is an fellow at the Potomac Institute of Policy Studies in Arlington, VA. He is also the director of liaisons at Aeolus Systems, an engineering services provider based in Clearwater, Florida and Nashua, New Hampshire.

Prior to joining GMU as an assistant professor in 1999, he was a senior systems engineer at Honeywell Space Systems in Clearwater, Florida. He has been a visiting post-doctoral fellow at the Army High Performance Research Center at the University of Minnesota, and an assistant professor of Mathematics at the University of Wisconsin. Wijesekera received a PhD in Computer Science from the University of Minnesota in 1997 under Professor Jaideep Srivastava and a PhD in Mathematical Logic from Cornell University in 1990 under Professor Anil Nerode.

James Bret Michael is a professor of computer science and electrical & computer engineering at the Naval Postgraduate School. His primary areas of research areas are engineering distributed and trustworthy systems. Prior to joining NPS, he conducted research at the University of California at Berkeley. He is the chair of the IEEE Technical Committee on Safety of Systems, a member of the Advisory Board for IEEE Software, an associate editor-in-chief of IEEE Security & Privacy, and an associate editor of the IEEE Systems Journal. He received his PhD in information technology from George Mason University. He is a senior member of the IEEE.

Man-Tak Shing is an associate professor of computer science at the Naval Postgraduate School. His research interests include software engineering, modeling and design of real-time and distributed systems, and the specification, validation, and runtime monitoring of temporal assertions. He is on the program committees of several conferences dedicated to software engineering and is a member of the Steering Committee of the IEEE International Rapid System Symposium. He was the program co-chair for the IEEE Rapid System Prototyping Workshop in 2004 prior to being the general co-chair for the symposium in 2008. He received his PhD in computer science from the University of California, San Diego. He is a senior member of the IEEE.